# Assignment No:5

Practical Exercise: Minimax Algorithm for Game Playing

## 1) Problem Statement

Implement the Minimax algorithm for a game-playing application such as Tic-Tac-Toe. The Minimax algorithm is used to choose the optimal move by simulating all possible moves and selecting the one that maximizes the player's chance of winning while minimizing the opponent's advantage.

## 2) Libraries Used

Python:  
1. **Math**: For representing infinity in the minimax algorithm.  
2. **Basic Python Data Structures**: Lists are used to represent the game board and manipulate game states.

## 3) Theory

The Minimax algorithm is a recursive method used in decision-making and game theory. It is designed to find the optimal move for a player assuming that the opponent also plays optimally. The algorithm evaluates the possible moves for both the player (maximizer) and the opponent (minimizer), and selects the move that maximizes the player's minimum gain.  
  
In a two-player zero-sum game like Tic-Tac-Toe, each player tries to either maximize or minimize the outcome. The algorithm explores all potential moves, assigns a score to each possible outcome, and chooses the move that provides the most favorable result for the player.

## 4) Methods

1. **Game Board Representation**: The game board is represented as a 3x3 grid where each cell can be empty ('\_') or occupied by a player ('X' or 'O').  
2. **Minimax Function**: This recursive function evaluates all possible moves by simulating future game states. The algorithm uses two parameters:  
 - `is\_max`: A boolean value indicating whether it's the player's or opponent's turn. The player tries to maximize the score, and the opponent tries to minimize it.  
3. **Game Evaluation**: The algorithm evaluates the current game state to check for a win, loss, or draw. If a win condition is met for the player, it returns +10, and if the opponent wins, it returns -10.  
4. **Base Case**: If no moves are left or the game has been won, the algorithm returns the appropriate score.  
5. **Optimal Move Selection**: The algorithm chooses the move that either maximizes the player's score or minimizes the opponent's score, depending on whose turn it is.

## 5) Advantages and Disadvantages

- **Advantages**: Minimax guarantees the best move for the player if both the player and the opponent play optimally. It provides a thorough evaluation of all possible game states.  
- **Disadvantages**: The Minimax algorithm can be computationally expensive, especially for games with larger state spaces (e.g., chess), as it explores all possible moves. Optimization techniques like alpha-beta pruning can help reduce computation time.

## 6) Diagram

## A screenshot of a computer Description automatically generated

## 7) Conclusion

The Minimax algorithm is a fundamental approach to game-playing AI, providing a strategic method to ensure that the player makes the optimal move assuming the opponent also plays optimally. While Minimax works effectively for small games like Tic-Tac-Toe, more complex games may require enhancements such as alpha-beta pruning to handle the exponential growth in possible game states.